R-D

INSTITUTE FOR COMPUTER APPLICATIONS IN

SCIENCE AND ENGINEERING (ICASE)

ABSTRACTS FOR THE

SECOND LANGLEY CONFERENCE ON SCIENTIFIC COMPUTING

NUMERICAL METHODS FOR PARALLEL AND VECTOR PROCESSORS

CO-SPONSORED BY ICASE AND SIAM

October 21 and 22, 1974

(NASA-TM-85171) ABSTRACTS FOR THE SECOND LANGLEY CONFERENCE ON SCIENTIFIC COMPUTING NUMERICAL METHODS FOR PARALLEL AND VECTOR PROCESSORS (NASA) 42 p

N83-72685

Unclas
00/64 08738

Abstracts for the

Second Langley Conference on Scientific Computing

Numerical Methods for Parallel and Vector Processors

Co-Sponsored by ICASE and SIAM

October 21 and 22, 1974

Author Index

Papers by the following authors are listed alphabetically by first author. For multiple author papers, the first author is given in parenthesis.

J. W. H. Liu

N. K. Madsen

S. G. Margolis

L. McCulley

A. K. Noor

M. C. Pease

W. G. Poole, Jr. (J. C. Knight)

J. R. Rice

R. F. Riesenfeld

A. H. Sameh

J. R. Schiess

D. Stevenson (D. E. Heller)

H. S. Stone

J. C. Thompson

J. F. Traub (D. E. Heller, H. T. Kung)

R. G. Voigt (J. C. Knight, J. J. Lambiotte, Jr.)

S. J. Voigt (A. K. Noor)

R. C. Ward

D. S. Watanabe (S. M. Yen)

J. Wirsching (T. Kishi)

W. A. Wulf

S. M. Yen

G. Zaher (L. McCulley)

Donald C. Adams
Mail Stop 215-1
U.S. Army Air Mobility R&D Laboratory
Ames Directorate
Moffett Field, California 94035

An Overrelaxation Algorithm for the ILLIAC IV

The potential speed increases of the ILLIAC IV make it a candidate for overrelaxation problems. The goal was to find a parallel overrelaxation algorithm that will converge as rapidly as the conventional algorithms (such as SOR and SLOR) for the two dimensional transonic airfoil problem.

The procedure followed was to first program the SOR and SLOR algorithms in FORTRAN for a CDC 7600. Each mesh point was updated as soon as a new estimate was calculated, which is the conventional procedure for serial algorithms.

The SOR algorithm was then altered such that a whole horizontal line of new mesh point estimates was calculated before updating was done. The SLOR algorithm was altered such that a new estimate for each point in the mesh was calculated before updating was done. These alterations simulated how the algorithms could be implemented on the ILLIAC IV. It was found that by using the SOR and SLOR algorithms in the serial mode, the optimal relaxation factor (alpha) was approximately 1.8. However, it was found that the optimal value of alpha for these two algorithms in the parallel mode was quite low. In fact, both algorithms (parallel mode) either converged very slowly or diverged for any value of alpha greater than or equal to 1.

The main problem with these two algorithms in the parallel mode is the following: the new estimate at any mesh point is a function of the four mesh points surrounding it. In the serial case, two of these points are current estimates and two are the result of the previous iteration. However, in the parallel case, only one point is current and three points are old.

The third method tried was the Skew algorithm which is essentially the SOR with a different ordering scheme. An iteration step consists of calculating new estimates of the mesh points along diagonal(s) rather than a horizontal or a vertical line. Utilizing the diagonals enables new estimate calculations to utilize two new and two old estimates of surrounding points. Since there is virtually no difference between the serial and parallel modes, only the parallel mode was considered.

The Skew algorithm was programmed in CFD for the ILLIAC IV as well as the CDC 7600. Using the CFDX translator, the CFD program was translated to serial FORTRAN and then run on NASA-Ames' IBM 360/67 to check the results of the CDC 7600 program. An actual run was successfully made on ILLIAC IV, whose results were verified by the CDC 7600 program.

The Skew algorithm gave results that were comparable to the SOR and SLOR algorithms in the serial mode.

Mats Apelkrans
Department of Computer Sciences
University College
Växjö, SWEDEN

A Highly Parallel Method for Elliptic Boundary Value Problems

In this paper we propose a method using the parallel-shooting technique to solve elliptic boundary value problems on a rectangle. To be more specific let

$$(1)\quad u_{xx} + f(x,y,u,u_y,u_{yy}) = 0$$

be an elliptic equation given on a rectangle in the xy- plane with boundary values $u = g$ given on the boundary. (1) is then approximated in the y- direction using finite-difference- quotients of order p. This gives rise to a system of ordinary differential equations

$$(2)\quad Y'' + \frac{1}{k^2} F(x,Y) = 0,$$

where the Y- vector contains approximations for the solution u at distinct y- levels, y_j say, for $j=1,2,\ldots m-1$. Special care must be taken in order to handle the boundary conditions. Next rewrite (2) into the following form

$$(3)\quad Y_i'' + \frac{1}{k^2} a_i(x,Y)\, Y_i = G_i(x,Y)\ ,\ i=1,2,\ldots m-1.$$

Given an initial approximation for u (found e.g. with the standard 5-point formula applied to a linearised version of (1)), we iterate in (3) according to the following formula

$$(4)\quad Y_i^{(r+1)''} + \frac{1}{k^2}\, a_i(x,Y^{(r)})\, Y_i^{(r+1)} = G_i(x,Y^{(r)})\ ,\ i=1,2,\ldots m-1,\ r=0,1\ldots$$

These scalar equations are then solved using the parallel- shooting technique in each subinterval. One should note that generally one has to compute or store intermediate values for $Y^{(r)}$. Let (4) be integrated with an initial- value method of order q (e.g. = p). These calculations can be executed in parallel for each equation in (4) and also in parallel for each subinterval. The (r+1)st iterate $Y_i^{(r+1)}$ is then computed as the solution to a tridiagonal system of order (2n-2). These (m-1) systems can be computed in parallel and there is also a bit of parallelism in a tridiagonal system (see eg Traub, Iterative solution of tridiagonal systems on parallel or vector computers, 1973). An obvious advantage of this method is that the resulting linear systems always are tridiagonal, independent of the approximation orders q and p . Furthermore, on a parallel- computer with enough CPU:s each iteration- step can be performed very quickly.

F. G. Carty
Goodyear Aerospace Corporation
Akron, Ohio 44315

Unconstrained Optimization Algorithms without Derivatives on the STARAN

This paper describes algorithms which locate the minimum of a function $f(x)$ in n variables and how their implementation can be enhanced by the capabilities of a parallel processor such as the Goodyear Aerospace Corporation STARAN*. A brief description of the STARAN architecture is given to aid those people not familiar with it. The algorithms under discussion are limited to those algorithms which employ only function evaluations and univariate searches in given directions, i.e. no derivatives need be computed. One such algorithm is described by D. Chazan and W. L. Miranker in "A Nongradient and Parallel Algorithm for Unconstrained Optimization" SIAM J. Control 8(1970) pp. 207-217. Their algorithm incorporates features in Powell's method and Zangwill's method. Another algorithm, based on Gram-Schmidt orthogonalization, is described where knowledge of the Hessian matrix of $f(x)$ is replaced by employment of univariate searches. This algorithm, when used on a parallel processor such as the STARAN, has the following advantages: (i) when $f(x)$ is a quadratic function with positive definite Hessian the algorithm converges (theoretically) to the minimum in a finite number of steps, (ii) when $f(x)$ is convex, $f(x)$ is reduced after each iteration unless at a minimum (iii) the best available approximation to the minimum is used at each step and (iv) more than n processors can be used if they are available. Numerical examples will be given.

---

* T. M. Goodyear Aerospace Corporation, Akron, Ohio 44315

Alan K. Cline
ICASE
NASA Langley Research Center
Hampton, Virginia 23665

A Lanczos Type Method
for the Solution of Large Sparse Systems of Linear Equations

A method for solving large, sparse, and unsymmetric systems of linear equations will be discussed. This method exploits an orthogonal bidiagonalization of the matrix and is a derivative of the Lanczos idea for orthogonal tridiagonalization of symmetric matrices. Operations are arranged so that only "operator knowledge" of the matrix is required. The decomposition and solution proceed in a column-wise fashion so that the only necessary storage is for several vectors of the same length as the solution.

The basic operations required by the algorithm are vector inner product, scalar times vector, and vector addition as well as the application of the linear operator and its transpose to a vector. All but the last of these operations are well suited for parallel or vector computation; efficient implementation on such computers requires that the last (i.e., application of the operator and its transpose) also be made efficient. The solution depends on the nature of the operator itself.

Various modifications to the basic algorithm have been explored. At each step more information of the decomposition is gained; the first modification determines the approximate solution yielding the minimal residual norm using this information. A second modification deals with blocks of vectors and a block bidiagonalization. This latter modification is found to maintain the orthogonality of the decomposition to a higher degree than the basic algorithm.

Richard L. Coleman
Digital Equipment Corporation
Maynard, Massachusetts 01754

Simulating the Performance of Gas Bearing Gyroscopes on the ILLIAC IV

Modern, high performance gyroscopes use gas bearings of various spiral groove configurations to support the spin axis member. The dynamic performance of these instruments is described by the time dependent Reynolds equation - a strongly nonlinear parabolic equation in two space variables. All modern designs use spiral groove geometries which in the asymptotic limit (as the number of grooves is large) by multiple scaling still yield the same type of equation for the mean pressure distribution. For the type of environment these gyroscopes are subjected to, it is necessary to solve for rapid pressure gradients both in space and time. The importance of the topic lies in the question as to whether an instrument can survive the expected shock and g loading. We have developed variants of fractional step methods with second order accuracy in time and at least second order accuracy spatially. Since these are locally one dimensional schemes, they can be efficiently solved for a given size problem on conventional computers; upon their introduction a factor of 10 improvement in running time was obtained. This now made feasible one degree of freedom simulations for gas bearing gyroscopes. Now, however, with the possibilities of parallel processors the locally one dimensional structure of these schemes can be fully utilized so that on a machine such as the ILLIAC IV the theoretical factor of 64 can be achieved. This now makes feasible the full five degree of freedom simulations of these gyroscopes - allowing us for the first time to correctly predict their actual performance. It will also be shown how on any parallel configuration with very modest memory such schemes can be applied. These techniques combined with a parallel processor allow certain classes of large scale simulations to be done which are of great importance in testing the feasibility of gyro designs where, because of size constraints, overdesign is not permissible.

George F. Corliss
Department of Mathematics and Statistics
University of Nebraska-Lincoln
Lincoln, Nebraska 68508

Parallel Rootfinding by Lagrange Interpolation

A class of rootfinding algorithms suitable for implementation on an array processor like ILLIAC IV using inverse Lagrange interpolation is shown to have a speed-up ratio proportional to the log of the number of processors used.

This agrees with the speed-up results found by W. L. Miranker in "Parallel Methods for Approximating the Root of a Function," IBM J. Res. Develop, 13(1967), pages 297-301. Miranker requires each processor to compute a Lagrange interpolation polynomial of a different degree, so his algorithm is not suitable for implementation on an array processor.

If each component of the vector $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,N})$ converges to α and $\delta_i = \max\{|\alpha - x_{i,j}|, j=1,2,\ldots,N\}$, then the sequence is said to have order of convergence λ if

$$\lim_{i\to\infty}(-\ell n\ \delta_i)^{1/i} = \lambda.$$

The parallel rootfinding algorithms considered are generalizations of the simultaneous n - 2 degree methods given by John R. Rice in "Matrix Representations of Nonlinear Equation Iterations - Applications to Parallel Computation," Mathematics of Computation, 25, 116(1971), pages 639-647. Assume that N processors are available. Let $X_0 = (x_{0,1}, x_{0,2}, \ldots, x_{0,N})$ be N initial approximations to the simple root α of f. Fix k satisfying $2 \le k \le N - 1$. Choose N distinct subsets A_j of the set $\{1,2,\ldots,N\}$ such that $j \in A_j$ and A_j contains k elements. Each processor constructs $H_j(y)$, the inverse Lagrange interpolation polynomial of degree at most k - 1 which satisfies

$$H_j(f(x_{i,s})) = x_{i,s}, \text{ for } s \in A_j \text{ and } j = 1,2,\ldots,N.$$

Define $x_{i+1,j} = H_j(0)$. If $f \in C^k$ in a neighborhood of α, then each component of X_i converges to α and the method has order of convergence k. The proof is a generalization of that given for the sequential case in Ralston's <u>A First Course in Numerical Analysis</u>.

We use the definition of efficiency for rootfinding algorithms due to H. T. Kung and J. F. Traub in "Computational Complexity of One-Point and Multipoint Iteration," Department of Computer Science, Carnegie-Mellon University, (1973):

$$\text{EFF} = \frac{\log_2 \text{ order of convergence}}{\text{number of parallel arithmetic operations per iteration}}.$$

Then we can choose the degree of the interpolating polynomials, depending on the number of arithmetic operations required to evaluate the function f, to optimize the efficiency. When this parallel efficiency is compared to the optimal efficiency of two classes of algorithms considered by Kung and Traub, the parallel rootfinding methods using inverse Lagrange interpolation are shown to have an assymptotic speed-up ratio proportional to $\log_2 N$ for functions which are very costly to evaluate.

John N. Damoulakis
Texas Instruments Incorporated
P. O. Box 2237
Huntsville, Alabama 35804

The Advanced Scientific Computer and its Application to Optimization Theory of Large Real-Time Systems

This paper examines the applicability of a vector processor machine to numerical methods in Mathematical Programming for large systems. A methodology utilizing the architecture and the vector programming features of the Advanced Scientific Computer (ASC) is developed by establishing suitable vectorial forms of numerical techniques to solve large optimal problems. The paper first examines the classical first-order methods in Mathematical Programming, and then derives a unified methodology for obtaining modifications of these methods, appropriate to this class of problems, for their efficient implementation in vector oriented machines.

Specifically, the general optimal problem

$$\text{Maximize:} \quad J = \prod_{i=1}^{i=N} [P_i(x_i)] \sum_{j=1}^{j=K} \left[\prod_{i=1}^{i=N_j} [R_{ji}(y_{ji})] \right]$$

$$\text{Subject to:} \quad \sum_{i=1}^{i=N} [c_i x_i] + \sum_{j=1}^{j=K} \sum_{i=1}^{i=N_j} [d_{ji} y_{ji}] \leq C^*$$

$$c_i \geq 0, d_{ji} \geq 0$$

$$0 \leq x_i \leq x_i^* \text{ ; and } 0 \leq y_{ji} \leq y_{ji}^*$$

is investigated, where the functions $P_i(x_i)$ and $R_{ji}(y_{ji})$ are nonnegative, concave, monotonically increasing and the constraints linear.

For the solution of this problem, first the applicability of the Marginal Return Optimization Method is investigated (MROM). In this method, an iterative algorithm is devised which solves the optimal problem with or without decomposition by a procedure altering one and only one variable at each iteration. The technique is especially useful in cases where a discrete solution of the optimal problem is sought. Then, the Gradient Method Techniques, adapted to these problems, are employed and the results are compared with the ones obtained from the Marginal Return Optimization Method.

Based on the properties of the objective function and the constraints, a combined algorithm is proposed utilizing the principles of Gradient Techniques and the concepts of MROM. The objective of this algorithm is to construct an efficient technique applicable to a continuous or discrete problem, and suitable for real-time operation, where computer execution speed is important. The algorithm does not decompose the original problem. It only utilizes the concavity condition of the functions to establish the necessary optimal criteria.

In order to establish the relative performance of these algorithms, numerical examples involving problems in resource allocation are provided. Particularly, the interest is in developing an effective operational system for the real-time environment. All algorithms are programmed in a way utilizing the maximum efficiency of ASC vector instructions. Typical problems involve a large number of functions, P_i, R_{ji} (between 30 and 150). Execution time measurements of these algorithms on the ASC are also provided.

James H. Ericksen
Laboratory for Atmospheric Research
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

The Implementation of a Two Dimensional Convection Problem on ILLIAC IV Using Disk

A program to solve the Bénard-Rayleigh convection problem on ILLIAC IV using disk is described. Two dimensional, 512 by 512 meshes are used. The problem is divided into two parts. First the prognostic equations are solved for temperature and vorticity. Then the stream function is calculated using the vorticity.

To solve the prognostic equations for time step $\tau + 1$, values on five meshes are required: the vorticity at τ and $\tau-1$, the temperature at τ and $\tau-1$, and the stream function at τ. The finite difference schemes used allow row I of the temperature or vorticity to be calculated whenever rows I-1, I and I+1 of the meshes are available. A queue is set up in memory which is divided into three sections. Each section contains eight rows of each of the five meshes. The program calculates eight rows in one section while it is writing from another section or reading from the remaining section.

Calculating the stream function is more difficult. Direct methods have been developed to solve Poisson's equation but they require the ability to access both rows and columns. Skewed storage is commonly used to access rows and columns of meshes contained in memory. By using this type of storage scheme on disk the I/O time can be decreased. No matter what storage scheme is used the data must be read and written to disk three times, once for the row operations, then for the column operations, and finally for the second set of row operations. As seen in Ericksen (p. 68) this portion of the code is quite I/O bound.

By using a method which doesn't require both row and column accessing from disk to solve Poisson's equation most of the I/O bound is eliminated. Hockney's Method with three levels of odd-even reduction converts the 512 by 512 element system of equations into one 512 by 64 element system and 448 systems with 512 elements each. The reduction into and the solution of the 512 element systems is performed along with the calculation of the prognostic equations. The 512 by 64 element system is solved without using disk. The data on disk needs to be read only once, when the prognostic equations are calculated.

The most difficult part of implementing Hockney's Method is finding a parallel method for solving a 512 element tridiagonal system. Buneman's method as described in Hockney (p. 155) is used. It is implemented in ASK with the ability to solve tridiagonal systems with 2^I-1 elements, where $2 \leq I \leq 12$. A storage scheme is developed for this method to allow the PE's to access 64 elements in parallel from a vector stored in PEM where the index for each element is $m(2^J)+n$. The J and the n are constant for the 64 elements but the m has 64 different values.

Ericksen, J. H., 1972: Iterative and Direct Methods for Solving Poisson's Equation and Their Adaptability to ILLIAC IV." Center for Advanced Computation, Document No. 60, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

Hockney, R. W., 1970: "The Potential Calculation and Some Applications." Methods in Computational Phyiscs, 9: 136-211.

Ogura, M., M. S. Sher, and J. H. Ericksen, 1972: "A Study of the Efficiency of ILLIAC IV in Hydrodynamic Calculations." Center for Advanced Computation, Document No. 59, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

Paul O. Frederickson
Lakehead University
Thunder Bay, Ontario

Fast Approximate Inversion of Large Elliptic Systems

We consider linear systems which are not only symmetric and positive definite, but also have the same sparse structure as the linear systems associated with the uniform partition discretization of boundary value problems in two or more variables. As the discretization is refined, and the order n of the system becomes large, solution of the system by the usual iterative methods such as SOR and its variants becomes increasingly slow and expensive, for their spectral radius tends to one. In contrast, the spectral radius of the algorithm we discuss has a bound independent of n and, according to empirical observation, in fact less than 1/2 for a large class of problems. The number of operations per iteration is linear in n, and the algorithm is so well adapted to parallel computation that the number of operations in an m-vector processor is bounded by a constant times $n/m + \log_2 n$.

The key step in the algorithm is a dyadic, or binary, collection algorithm with binomial coefficients as weights. In effect, a sequence of systems of lower order is initially constructed and then repeatedly solved, approximately. These are best approximate systems in the variational sense, for the exact solution to any would be the best approximate solution to any finer system after spline interpolation.

The algorithm is already effective on seriel computers, as several months of testing on the Telefunken computer of the Leibniz Rechenzentrum in München have shown. For example, 14 seconds per pass were required with $n = 4225$, which is about the point at which this algorithm becomes competitive on seriel computers of current type. The advantage will be clearer on vector processors.

Linda J. Hayes

Center for Numerical Analysis
University of Texas
Austin, Texas 78712

Texas Instruments
P. O. Box 2909
Austin, Texas 78767

Timing Analysis of Iterative Techniques for Solving Linear Systems of Equations on the ASC

With the development of parallel processors, questions have arisen concerning which serial algorithms are directly transferable to parallel processors, which are not, and why some are better suited for parallel processing than others. This presentation considers these questions with respect to iterative methods for solving systems of linear equations where the matrices are large and sparse, such as those which arise in the solution of elliptic partial differential equations by finite difference methods based on the five point formula. The algorithms considered are Jacobi, Gauss-Seidel (both with natural and with red-black orderings), Successive Over-relaxation (SOR) (both with natural and with red-black orderings), Sheldon with red-black ordering, Cyclic Chebyshev Semi-iterative (CCSI), Jacobi Semi-iterative, and Symmetric Successive Over-relaxation (SSOR-SI). Numerical experiments were performed using these methods with the "model problem" involving Laplace's equation on the unit square. A primary objective was to determine the effect of the non-parallel sequences of code upon the execution times of the algorithms. Timing reults are presented for all nine methods with mesh sizes of 1/20, 1/40 and 1/80 corresponding to 361, 1521 and 6241 data points. These methods were coded, executed and timed on the Texas Instruments' pipeline processor, the ASC, located in Austin, Texas.

With the natural ordering the Gauss-Seidel, SOR and SSOR-SI methods have a non-parallel sequence of code which must be performed separately for each data point, allowing the pipeline to completely clear on a pipeline processor. On an array processor such as the Illiac IV, this sequence of code would allow only one processor to be active. This situation severly degrades execution speed of these algorithms on parallel processors. Timing results on the ASC show that these three methods require between 100-700 percent more execution time per iteration than the other six methods tested. Techniques are suggested which would reduce the non-parallel structure of these algorithms, thus increasing their speed. These techniques include re-ordering of mesh points, rearrangement of data in memory and use of block SSOR-SI methods which can take advantage of parallel techniques for solving tridiagonal systems.

Don Heller
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems

We consider the solution of the block tridiagonal linear system $Ax = v$,

$$e_j x_{j-1} + d_j x_j + f_j x_{j+1} = v_j, \quad j = 1,\ldots,N,$$

$N = 2^{m+1} - 1$, $m \geq 1$, $e_1 = f_N = o$, where the components are $n \times n$ matrices and n-vectors. By a cyclic reduction algorithm we generate a sequence of problems $A^{(i)} x^{(i)} = v^{(i)}$, $i = 1,\ldots, m$, of block dimension $(2^{m-i+1} - 1) \times (2^{m-i+1} - 1)$, with $x_j^{(i)} = x_{2^i j}$. The solution to system i is used in a back substitution to obtain the solution to system $i-1$. Let $y^{(k)}$ be an approximation to $x^{(k)}$, and let y be an approximation to x obtained by back substituting $y^{(k)}$. Let $B^{(i)}$ be the matrix associated with the block Jacobi iteration for $A^{(i)} x^{(i)} = v^{(i)}$, $A^{(o)} = A$. $\| B^{(i)} \|_\infty$ measures the diagonal dominance of $A^{(i)}$. With this notation we have the following theorem:

Theorem. Suppose $\| B^{(o)} \|_\infty < 1$. then

1. the cyclic reduction algorithm is well-defined.
2. $\| x - y \|_\infty = \| x^{(k)} - y^{(k)} \|_\infty$.
3. $\| B^{(i+1)} \|_\infty \leq \| (B^{(i)})^2 \|_\infty$, $i = o,\ldots, m-1$.

This effect of quadratic convergence allows early termination of the reduction (i.e., $k < m$) when an approximate solution is acceptable and N is large.

D. Heller, D. Stevenson, and J. F. Traub
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Iterative Solution of Tridiagonal Systems Using Red-Black Order

We study the parallel iterative solution of tridiagonal systems. Two types of systems are analyzed. The first type are based on additive decomposition. Examples are SOR and Two-Cyclic Chebyshev iterations. The second type are based on iterative multiplicative decomposition.

The methods are compared for a number of models of parallel computation. The results of numerical experimentation are also reported.

L. Hyafil
IRIA-LABORIA
78150 Rocquencourt
France

H. T. Kung
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Parallel Algorithms for Solving Triangular Linear Systems

The problem of solving triangular linear systems of n equations on parallel computers such as the ILLIAC IV is considered. Assume that each arithmetic operation takes one unit of time. From previously known results, we know that with $O(n)$ processors the problem can be solved in time $O(n)$ and with $O(n^3)$ processors the problem can be solved in time $O(\log^2 n)$. In this paper we ask the following question: How fast can the problem be solved with $O(n^a)$ processors, where $0 < a < 3$? We show that with $O(n^a)$ processors the problem can be solved in time $O(n^{2-a} \log n)$ if $0 < a < 1$ and in time $O(n^{1-a/3} \log^2 n)$ if $3/2 \leq a < 3$. (See the table below.)

The basic technique used in this paper is a reduction technique which reduces parallel algorithms with large parallelism to parallel algorithms with small parallelism. We believe that similar techniques can be used for many other interesting problems.

Number of Processors	Upper Bounds on Time
1	$O(n^2)$
$O(n^{1/2})$	$O(n^{3/2} \log n)$
$O(n)$	$O(n)$
$O(n^{3/2})$	$O(n^{1/2} \log^2 n)$
$O(n^2)$	$O(n^{1/3} \log^2 n)$
$O(n^{5/2})$	$O(n^{1/6} \log^2 n)$
$O(n^3)$	$O(\log^2 n)$

Thomas L. Jordan
Los Alamos Scientific Laboratory
Los Alamos, New Mexico 87544

Some Considerations in Implementing Gaussian Elimination for Dense Systems on STAR and a New Parallel Algorithm for Diagonally Dominant Tridiagonal Systems.

The problems of implementing Gaussian elimination for dense linear systems on the STAR-100 is considered. First we discuss properties of the STAR-100 that have the most influence on algorithm performance and then three methods for implementing Gaussian elimination are discussed. These methods differ in storage assignment, indexing techniques, etc., and both positive and negative features of each are discussed. The selected method has an additional feature in that arithmetic cancellation can be monitored dynamically within it.

A new parallel algorithm is given for solving diagonally dominant tridiagonal systems. The algorithm requires $\log_2 n$ parallel steps where each parallel step consists of 8 multiplies, 4 adds, and 1 divide of vectors of average length almost n. However, in the case of strict diagonal dominance, the algorithm may converge in fewer than $\log_2 n$ steps. Hence, the process may be truncated. Estimates of the STAR-100 time required for systems of various orders will be made.

Codes are available for the general linear system, the dense symmetric positive definite case and the diagonally dominant tridiagonal system. These codes are written in standard FORTRAN and have been checked out on the CDC-7600. The vectorizable portions of the codes are identified and are written so that a translation to the vector syntax of the STAR CDD-FORTRAN is trivial. In each code all floating point arithmetic is implementable with STAR-100 vector instructions.

Ronald Karn
Computer Sciences Corporation
2880 Broadway
New York, N.Y. 10025

GISS Model Illiac Implemantation

The Goddard Institute for Space Studies (GISS) is rewriting and test-running its Nine-Level Atmosphere General Circulation Model on the Illiac IV so as to utilize the machine's parallel/vector capability. The GISS model, which currently runs on a 360/95, is strongly parallel in the dimension of longitude. This project will enable GISS to assess the value of parallel processing as a solution to its computing needs.

Utilization of the Illiac posed the following key technical Problems:

1. The relatively small core capacity of the Illiac IV (as opposed to the 4500K user-available bytes of the 360/95) required a redesign of the program in order to utilize the disk as a core extension.
2. It was necessary to perform the initial testing with a core-contained version, since the Illiac disk I/O hardware had not yet been fully debugged.
3. To produce acceptable resolution, the GISS model required more longitudes than there are processing elements in the machine.

The GISS solution to problems 1 and 2, which is applicable to any two-time-step differencing scheme in which the space differencing involves only neighboring values, enables the model to run core-contained with 40% less core or, alternatively, to run with disk extension and 77% less core. Problem number 3 could not be solved merely by successively executing the code for two or more segments of the earth's circumference, because the dynamics of the atmosphere induce a strong east/west interaction. In a computer implementation this interaction corresponds to a circular routing with wraparound, which is non-trivial for vectors of length greater than 64. Each east/west interaction was therefore implemented in two stages, as follows:

1. Rotating one argument of the interaction by means of a special-purpose subroutine to bring it into spatial correspondence with the other argument.
2. Carrying out the necessary computations in sequence for each 64 P.E. segmant of the two arguments.

As of this writing, (6/10/74) GISS has debugged and run a core-contained version of the model with 64 longitudes, 31 latitudes and 9 pressures. This version contains a complete set of hydrodynamic routines, which are concerned with the continuity and conservation properties of the atmosphere and which require the bulk of the CPU time and the memory space in a computer implementation. It does not contain routines for physically more complicated and relatively lower magnitude processes such as surface friction, solar radiation, convection, etc. Coding on these "physics" routines and on extending the grid is well advanced. With the core-contained version initial measurements show a factor of 6 speed improvement over a comparable version on the 360/95.

Tadashi Kishi and J. Wirsching
Lawrence Livermore Laboratory,
University of California
Livermore, California 94550

Vectorization of Particle-Pusher Codes

This paper addresses itself to a vectorized computational procedure that is applicable to the class of particle simulation models that are critically important in future controlled fusion research. Historically, plasma physics codes have evolved from small grids and simple physics to larger and larger grids with more complex physics. Currently, particle-pusher models, which follow a large number of charged particles under the influence of self-consistent electric and magnetic interactions, already tax the resources of such computers as the CDC 7600 and IBM 360/195. Until now, not much attention has been given to the computational procedure because the computer speeds have been quite adequate. With the advent of the highly parallel "vector" type computers such as the ILLIAC IV, STAR, and ASC a new problem has arisen: The computational procedure currently in use is not easily or efficiently adaptable to this class of machines. It is our intention to describe a method that may be efficient for "vector" solution of plasma and particle in cell codes.

The computational procedure of particle-pusher codes involves three phases: (1) the motion of particles dependent on local electrostatic and magnetic field quantities, (2) updating of the charges distribution resulting from the motion of the particles, and (3) solution to Poisson's equation of the new charge distribution to provide new magneto-static quantities. The problem that occupies a great portion of the "particle-pusher" type code is basically a table look up procedure in which the values to be retrieved are found to be essentially random in their location in the table. To a serial computer this is no drawback, but to a vector computer the process is highly inefficient. Although the table look up has even been put into hardware on the STAR, this does not ensure high performance.

The answer to the dilemma is to develop, if possible, a way of efficiently creating contiguous vectors out of random processes. The method described in the report attempts to do this. The factors that help in our proposed procedure depend first, on such concepts as sparce-vector operations, control bit vectors, and compress which are common to the STAR, ASC, and ILLIAC IV computer systems. Second, rather than process from the particle point of view, if we consider a given cell and its associated set of particles, a particle will not skip entirely over a neighboring cell in one time step and the order in which the particles are kept or processed is immaterial.

The problem is the accumulation of a charge at the grid points. Thus, if any success is to be gained in vectorizing the process, all of the particles in each cell must be made into a contiguous vector. There are certain trade-offs that must be made since this method introduces processing that a serial computer does not have to do. There are some mitigating circumstances even there, in that serial machines are heavily engaged in index and address arithmetic which would be obviated by the method described.

John C. Knight
Analysis and Computation Division
NASA Langley Research Center
Hampton, Virginia 23665

Vector Processors:

Control Data Corporation's STAR-100 and

Texas Instruments' Advanced Scientific Computer

These machines are termed vector processors because their hardware is especially designed to enable them to perform vector arithmetic at relatively high speed. At the heart of both are segmented execution units, and these are augmented with sophisticated main storage capable of considerable bandwidth. In anticipation of scientific workloads involving large amounts of data, high performance backing storage devices have been developed to support these systems. The concept of distributed computing, whereby several powerful satelite processors take over many of the routine support tasks, is exploited heavily by both machines.

A segmented execution unit, or "pipeline", is constructed as several subunits which are organized as a production line. At any given time, each subunit may be operating on a different pair of operands, but the desired result only becomes available when a given pair of operands has passed through all the subunits. Thus, a single operation may take several time units to complete, but a sequence of similar operations can be performed rapidly. Usually, one result is produced per time unit once the first result leaves the pipeline. Much higher result rates are possible with this technique than are possible with a conventional execution unit.

The key to the effective use of such hardware is the concept of vector instructions. A vector is a sequence of like objects (e.g., floating point numbers), and a vector instruction initiates a sequence of like operations. Both CDC's STAR-100 and TI's Advanced Scientific Computer employ segmented execution units and provide vector instructions. The instruction sets include the commonly occurring monadic and dyadic arithmetic operators, and several other operators which are usually only found in software.

The STAR-100 is equipped with hardware to deal directly with sparse vectors. A sparse vector is stored as a set of nonzero elements together with a bit vector. In addition, the elements of the result vector produced by many instructions can be stored selectively under control of a bit pattern known as a control vector.

The ASC has the useful capability of being able to encompass three dimensions of addressability in a single vector instruction. This is the equivalent of a nest of three indexing or "DO" loops, and allows matrix multiply to be written as one instruction.

John C. Knight
Analysis and Computation Div.
NASA Langley Research Center
Hampton, Virginia 23665

William G. Poole, Jr.
ICASE and
College of William and Mary
Williamsburg, Virginia 23185

Robert G. Voigt
ICASE
NASA Langley Research Center
Hampton, Virginia 23665

Configuration Analysis of the STAR-100 Based on Typical Numerical Problems

The STAR-100 central processor is capable of result rates which are considerably in excess of those of presently available computers. In order to be a competitive machine for scientific computing the STAR-100 must be able to sustain this result rate on very large data bases. This raises the questions of (a) whether the result rate will be severely limited by the ability of the peripheral devices to support the central processor, and (b) what data organizations are most suitable for secondary storage.

In order to provide some insight into these questions a problem was selected which is representative of large scientific computing. In particular we examine the solution of a banded system of linear equations using a vectorized version of Cholesky's algorithm. We also assume that there are at least an order of magnitude more non zero elements than main storage locations. Two approaches to this problem were compared: (a) treating the banded system in its entirety, (b) using the concept of substructuring or, equivalently, block decomposition in which several smaller banded systems are considered.

Timing of the algorithms shows that the input/output (I/O) requirements are considerably greater than the capabilities of existing rotating storage devices assuming a random access pattern. However, careful data layout so as to eliminate rotational delays and seek times of available discs leads to a balance of I/O and computation in the decomposition steps of the algorithms. On the other hand, even the optimum layout leaves the forward and back substitution phases I/O bound.

H. T. Kung
J. F. Traub
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Methodologies for Studying the Speed-Ups Gained from Parallelism

Let $\mathbb{P}$ be a class of problems to be solved and for each $p \in \mathbb{P}$ let $\lambda(p)$ be the probability that we want to solve p. In this paper methodologies for measuring the expected gain in speed due to parallel processing, for a class $\mathbb{P}$ with a probability function λ, are investigated.

For each $p \in \mathbb{P}$ and each positive integer k, let $T_k(p)$ be the minimum time needed to solve p with k processors. The following two measures of speed-ups gained from parallelism are proposed:

$$SA_k(\mathbb{P},\lambda) = \frac{\sum_{p\in\mathbb{P}} \lambda(p)T_1(p)}{\sum_{p\in\mathbb{P}} \lambda(p)T_k(p)},$$

$$AS_k(\mathbb{P},\lambda) = \sum_{p\in\mathbb{P}} \lambda(p)\frac{T_1(p)}{T_k(p)}.$$

It is shown that both measures are reasonable. The use of any particular measure depends on the application. However, only the second one is suitable for measuring the asymptotic gain in speed as $k \to \infty$. To illustrate our methodology, we show that for the polynomial evaluation problem the speed-up according to the second measure can be either $O(\log k)$ or $O(1)$ as $k \to \infty$, depending on the probability function.

Jules J. Lambiotte, Jr.
Analysis and Computation Division
NASA Langley Research Center
Hampton, Virginia 23665

Band and Profile Factorization Methods on the STAR-100 Computer

For the symmetric positive definite matrix A, where A has sparsity structure suitable to band or profile storage schemes, the factorization as $A=LDL^T$ for D diagonal and L unit lower triangular is considered for the STAR-100 vector computer. For A banded, the vector implementations of the Root-Free Cholesky (RFC) method and the Symmetric Gaussian Elimination (SGE) method are discussed. The desire to perform vector operations within the solution process greatly influences the order in which the coefficients within the band are stored. It is shown that two different storage strategies are required for the two methods, and that they differ from the usual strategy on a serial computer. Timing information for the STAR-100 is used throughout, and it is shown that the RFC method is faster than the SGE method for matrices with small bandwidths, but slower for larger problems. The vector implementation of the two substitution processes, $Ly=b$ and $L^Tx=D^{-1}y$ to solve $Ax=b$, is discussed and it is shown that different computational procedures are required for the two substitutions due to storage considerations. Also included is the analysis of the substitution procedures when a large number of right-hand sides exist.

The profile storage and computation scheme offers an alternative to band methods to take advantage of certain zero coefficients which may exist within the band. The vector implementation of this method on the STAR-100 using an inner product formulation analogous to RFC is considered. The banded and profile implementations are compared for a model finite element mesh and it is shown that except for very simple elements, the profile approach is expected to be faster. However, due to vector startup time, only as the number of elements in the mesh goes to infinity do we achieve the total gain in speed predicted for the profile method over the banded method by the serial operation count.

Jules J. Lambiotte, Jr.
Analysis and Computation Division
NASA Langley Research Center
Hampton, Virginia 23665

An Implementation for the ADI Method on the STAR-100 Computer

The alternating direction implicit method (ADI) for solving elliptic partial differential equations presents computational advantages on a serial computer since, for a model nxn mesh, each half iteration involves the solution of n tridiagonal systems of size n, a task which can be accomplished efficiently on such a computer. However, it has been shown that the usual solution process (Gaussian Elimination) for a tridiagonal system of equations does not lend itself to vector computation, and that none of the proposed parallel algorithms show a substantial increase over a scalar implementation on STAR until the size of the system becomes large. In addition to this, there is an added difficulty for implementation of the ADI method on STAR. That is, given a particular solution algorithm, a particular direction of implicitness, and the appropriate ordering of the mesh points to induce the desired storage for the solution algorithm, then when the direction of implicitness is alternated, the opposite ordering of the mesh points is required for that algorithm. This involves the transpose of the solution matrix of the grid points which is a slow operation on STAR even if all the data is core contained. If not all the data is core contained, it becomes considerably more inefficient. In order to avoid the necessity for the transpose operation, a mixed solution algorithm is proposed consisting of a different solution approach for each direction of implicitness. Two vector algorithms for solving α independent nxn tridiagonal systems are discussed. The first considers the systems as stacked one behind the other and solves all α at the same time using the usual serial algorithm but on vectors of length α. The second consists of adding zero coefficients to artificially couple the α systems to construct one new tridiagonal system of size $N=n\alpha$. Odd-even reduction, which is known to be efficient for matrices with large N, is employed to solve the system and it is shown that, in addition, only $\log_2 n$ steps are required in the factorization instead of the expected $\log_2 N$. It is then shown that the relationship of the required direction of storage to the required direction of implicitness for the two algorithms is such that it is no longer required to transpose the solution matrix.

Jules J. Lambiotte, Jr.
Analysis and Computation Division
NASA Langley Research Center
Hampton, Virginia 23665

Robert G. Voigt
ICASE
NASA Langley Research Center
Hampton, Virginia 23665

The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer

The usual method for solving tridiagonal linear systems requires one to evaluate simple recurrence relations. It turns out that evaluating such relations on the STAR-100 is not an optimal use of the machine since it is not possible to use the vector instructions provided by the hardware. Consequently it is desirable to consider other solution techniques.

We consider both direct and iterative methods and compare these with the usual method mentioned above. The direct methods include one proposed by H. Stone, a variation on his method, and cyclic reduction. These all allow the use of vector instructions with varying degree of success. However, we note that Stone's algorithm requires $O(n\log_2 n)$ arithmetic operations for a system of size n whereas the usual algorithm and cyclic reduction require only $O(n)$. It is shown that on a computer like the STAR-100 this difference is significant.

By placing conditions on the linear systems such as diagonal dominance we may consider iterative methods. We compare a parallel algorithm introduced by J. Traub with Jacobi's method, SOR, and the best of the direct methods, cyclic reduction. The conclusion is that the direct method is, in general, superior.

The comparisons made in this paper are based on codes written at the hardware instruction level and timed using the latest timing information for those instructions supplied by Control Data Corporation.

Joseph W.H. Liu
Department of Applied Analysis & Computer Science
University of Waterloo
Waterloo, Ontario, Canada

THE SOLUTION OF MESH EQUATIONS ON A PARALLEL COMPUTER

This paper discusses the applicability of "nested dissection" in the parallel solution of mesh systems by symmetric elimination. In our analysis, inherent parallelism is exploited on a computer model that is assumed to have an unlimited number of arithmetic processors. Each processor can perform any one of the binary operations in unit time. They obtain their instructions from a single instruction stream, so that they execute this same instruction, but on different operand pairs (single instruction stream, multiple data stream).

For systems associated with an n by n regular mesh, it is known that elimination using a nested ordering, which requires $O(n^3)$ multiplications and $O(n^2 \log_2 n)$ storage locations on sequential computers, is optimal. In this paper, we show how the dissection techniques can speed up the computation in its direct solution on our hypothetical parallel computer. A mesh dissection divides the mesh into several disconnected components through some separating set of nodes. Parallel elimination can then be performed on the independent matrix blocks corresponding to the components. The factorization process will be completed by eliminating the nodes in the separating set. By a recursive use of this idea, it is shown that nested dissection when coupled with parallel elimination requires $O(n)$ arithmetic operations to factor an n by n mesh system. Parallel substitutions of independent blocks allow the final solution to be obtained in again $O(n)$ operations.

We also introduce the doubly nested dissection technique, which uses two "layers" of separating nodes as the dissector. In the context of parallel elimination and substitution, it turns out to be a more practical way of dissecting the mesh since doubly dissection enables blocks to be factored and substituted in a completely independent manner. Again, $O(n)$ parallel operations are required for the complete solution by doubly nested ordering.

Niel K. Madsen
Lawrence Livermore Laboratory
Numerical Mathematics Group
Livermore, California 94550

Numerical Solution of Parabolic Partial Differential Equations on a Vector Processor

In this paper we consider some of the problems associated with obtaining efficient numerical solutions to a general class of parabolic partial differential equations in two space dimensions. In the past, it has been true that for most parabolic problems, implicit time discretizations were a necessity. The stability restrictions on time step sizes imposed by the explicit methods were simply so severe that these methods are not extensively used for parabolic problems.

On a vector processor such as the STAR-100, the explicit methods are very appealing because they may usually be implemented and run in an easy and efficient manner. In contrast, the implicit methods require the solution of a system of linear equations and are thus not as easy to efficiently implement on the STAR-100.

Much research is currently taking place to find appropriate methods for solving general systems of linear equations. However, the answers to this problem are as yet unclear. With the efficient solution of these general linear systems somewhat in doubt, the question of explicit versus implicit methods must again be reopened. We consider this question and conclude that the class of problems which may be effectively solved on vector processors by use of explicit methods has been broadened but still does not include many types of problems that are frequently encountered.

Finally, we describe the structure of a general purpose program designed for the STAR-100 which solves a fairly broad class of parabolic partial differential equations in two space dimensions. The program has some unique features such as variable order implicit time integration methods, automatic time step selection, time integration error controls, and is very easy to use and quite reliable.

Stephen G. Margolis
ASEA-ATOM Västerås Sweden
and State University of New York
Buffalo, New York 14226

Computing One-Dimensional Fluid Dynamics on Parallel Processors

Assessment of the safety of nuclear power plants requires the solution of the equations of one-dimensional fluid dynamics over a variety of time scales, ranging from milliseconds to hours. Both explicit and implicit methods have been used. The explicit methods are limited to time steps of milliseconds by the Courant-Friedrichs-Lewy Condition. Thus use of such methods to calculate phenomena lasting hours is extremely costly. Implicit methods are not limited to short time steps, but require rather longer computing times per time step because of the required solution of linear systems, which are often block tri-diagonal. Thus the use of implicit methods for problems on short time scales is inefficient. It is desirable that a single method be used, however, since often the same problem must be examined on both long and short time scales. Motivation then exists to find accurate and efficient implicit methods. H. S. Stone has recently shown that tridiagonal systems can be solved efficiently on parallel processors. (H. S. Stone, "An Efficient Parallel Algorithm for the Solution of a Tridiagonal System of Equations", J.ACM 20, 27 (1973)). With this development, implicit methods become practical for problems on both long and short time scales. In this paper, we describe a particular implicit method and test it for its accuracy in a shock-wave calculation. A generalized Crank Nicholson Method is used, in which space-derivatives are approximated by a weighted average of space differences at "old" and "new" times. Best results are obtained when the "new" space differences are given a weight of 0.7. Momentum flux is approximated using a non-linear method (ZIP differencing) which has advantages in symmetry and accuracy over other methods. Results indicate that the method is sufficiently accurate, and with the availability of parallel processors ought to be sufficiently fast, to permit use of implicit methods on all time scales.

L. McCulley and G. Zaher
Informatics Inc.
3971 East Bayshore Drive
Palo Alto, California 94303

Heat Shield Response to Conditions of Planetary Entry Computed on the ILLIAC IV

Temperature within a reflecting, ablating material subject to convective and radiative loading satisfies in one dimension a non-linear diffusion equation of the form $T_t = a(x, T)\, T_{xx} + b(x, T)\, T_x + c(x, T)$. Quantitative correlations of loading to material behavior at the surface are obtained from a preliminary shock layer calculation for simulated entry into a model atmosphere. The surface boundary condition for the diffusion equation is derived from these correlations. Subsurface absorption of radiation is expressed by the distributed source term $c(x, T)$. Temperature profiles are obtained numerically at discrete points in time by solving the penta-diagonal linear system resulting from second order finite difference approximation of the diffusion equation at each of sixty-four non-uniformly distributed mesh points. Three candidate methods for solution of this linear system are implemented and evaluated: (1) a direct method derived as an extension of H. S. Stone's method for tri-diagonal systems (JACM, Vol. 20, No. 1, Jan. 1973, pp. 27-38), (2) Jacobi's iterative method, and (3) a modified Jacobi method which converges more rapidly. The direct method proves to be least efficient of the three. The iterative methods are comparable in this application, with each exhibiting superiority at different points within a single case run. A further consideration is the need to solve a system of radiative transfer equations for evaluation of subsurface absorption. Because scattering and absorption for the shielding material considered here are temperature and frequency dependent, the usual analytic solution is not available and these equations must be solved numerically for each of the twenty radiation bands important in this problem. Numerical formulation of the problem gives rise to a block (2 x 2) tri-diagonal matrix equation. Solution by parallel methods similar to those above is discussed. It is shown however that simultaneous solution for all bands, by assigning individual processing elements the common serial task of solution for a single frequency, is more efficient.

Ahmed K. Noor
George Washington
University Center

Susan J. Voigt
Structures & Dynamics Division

NASA Langley Research Center
Hampton, Virginia 23665

A Hypermatrix Scheme for a Finite Element System on the STAR-100 Computer

In applying the finite element method to analysis of complex structures, the resulting large matrices do not fit completely in the core storage of the computer. It is necessary, therefore, to subdivide these matrices to perform the calculations. In studying the potential use of the CDC STAR-100 computer for solution of structural analysis problems, the hypermatrix scheme appears to be especially well-suited. The hypermatrix (or block matrix) technique is based on partitioning the structural matrices in both the row and column direction. Then, operations are performed on the submatrices. An address (or pointer) matrix is used to identify the locations of the different elements (e.g., submatrices) in the hypermatrix. A zero entry in the address matrix denotes a zero submatrix which is neither built nor stored; i.e., the hypermatrix scheme provides a macro representation of the sparsity pattern. The subdivision of the matrix may follow physical reasoning, or it may be selected according to other considerations. For example, one submatrix may contain all degrees of freedom of a particular node, element, or a substructure; or subdivision may follow from consideration of the sparsity pattern of the matrix to use central memory efficiently for a specific problem.

The hypermatrix scheme has been used on current scientific computers combining a number of advantages over other techniques of handling large systems of equations. It provides an effective way of using the backing store for management of the large quantities of data in finite element systems. It allows a controllable ratio between CP and I/O times, and it provides versatility in the sense that the core storage requirements are independent of the problem type and size. The scheme has been incorporated into a number of large general purpose finite element structural analysis computer programs including ASKA, DAISY, and SESAM-69.

The objectives of this paper are to study the adaptation of the hypermatrix scheme to the CDC STAR-100 computer and to examine alternative strategies for exploiting the special capabilities of the STAR computer. The restricted locality of data reference inherent in the hypermatrix scheme appears quite attractive for the effective use of the virtual memory facility. Discussion is focused on the organization of the hypermatrix computation and the mode of storage of the different submatrices to take advantage of the STAR pipeline (streaming) capability using the Gaussian elimination and Cholesky decomposition procedures. Consideration is also given to associated data handling problems and means for reducing the page faults in the solution process. Results of operation counts are presented showing anticipated gain in speed over the CDC 6600 to be obtained by using the proposed algorithms on the STAR computer.

Marshall C. Pease
Computer Science Group
Stanford Research Institute
Menlo Park, California 94025

A New Algorithm for Linear Equations and Matrix Inversion

A new algorithm for solving linear equations or for the inversion of a matrix is presented. The algorithm is essentially parallel in nature. The basic operation is the component product of two vectors, which is the vector whose k-th component is the product of the k-th components of the two vectors. Hence the algorithm is well adapted for execution on a parallel computer such as ILLIAC IV, or on a machine constructed as a network of microcomputers, where the network connections and control are of the type used in ILLIAC IV.

If the efficiency of the algorithm is measured by the count of individual multiplications without regard for parallelism, it is less efficient than Gauss' algorithm, but not seriously so. The ratio of the count of multiplications to that of Gauss' algorithm is approximately $(1 + \log_2 n)$ for matrix inversion, and $(3/4) \log_2 n$ for linear equations, if the matrix is $(n \times n)$.

The generality of the algorithm is not complete--there do exist non-singular matrices that cannot be inverted by the algorithm since one or more of the operators used are singular. In principle even these cases can be handled with a suitable variation of the algorithm, but there seems to be no practical way of discovering what is the appropriate variation. Neither does there seem to be any practical way of identifying, a priori, those matrices for which it will fail. Fortunately, such a failure happens only under pathological conditions with vanishingly small probability of occurrence for matrix classes that are likely to be important. Further, if a failure should occur, this fact would become evident almost immediately. It does not lead to an incorrect solution, but rather to the absence of any solution. Hence the possibility of failure of the algorithm does not seem likely to interfere with its practical use.

The significance of the algorithm is that it expands the range of applications for which the specified type of parallel architecture can be used effectively. Such architectures may be said to address directly such problems as the solution of partial differential equations under assumed boundary conditions. The algorithm presented here makes that type of parallel architecture useful also for applications requiring the solution of linear equations or matrix inversion.

John R. Rice *
Division of Mathematical Sciences
Purdue University
West Lafayette, Indiana 47907

Parallel Algorithms for Adaptive Quadrature II - Metalgorithm Correctness

A metalgorithm (or perhaps metaprogram) represents a class of algorithms and previous papers have established convergence results for sequential [1] and parallel [2] adaptive quadrature computations. These results may be paraphrased as follows: "If all the algorithms represented by the metalgorithm satisfy certain assumptions, then a certain rate of convergence takes place." The assumptions are fairly simple in nature and the results are typical mathematical convergence theorems. They do establish the exceptional power of adaptive quadrature to handle badly misbehaved integrands without loss of efficiency.

The utlimate goal is to prove convergence for an actual computer program and this paper represents the second level of analysis. A much more concrete metalgorithm is described and it is proved that it is contained in the earlier, more general metalgorithm. This metalgorithm involves NCPU + 2 central processors, NCPU of which execute basic quadrature computations. The other two represent the operating system and the algorithm controller. The metalgorithm contains 9 programs and a variety (38 in all) of attributes are assumed about these programs. The metalgorithm also contains assumptions about the allocation of memory and the timing of various processes (it is an asynchronous, multiple-instruction-stream, multiple-data-stream computation). Once we prove that this metalgorithm is contained in that of [2], then we have the convergence result established; a subsequent paper will contain a program PAFAQ and a proof that it is represented by the present metalgorithm.

The present metalgorithm is detailed enough that the mechanisms are presented for preserving the integrity of the data structures in the concurrent programming environment. A key point in the third level of proof involving PAFAQ will be to show that these mechanisms are correctly implemented. An unusual property of this metalgorithm is that algorithmic convergence rather than mathematical convergence is established. That is, any algorithm represented by this metalgorithm has an accuracy requirement as input and the algorithm will terminate and print a quadrature estimate that meets this requirement. This is achieved by having a characteristic length of the integrand as input data. The role of the characteristic length of a function (relative to an algorithm) in proofs of algorithm effectiveness is introduced in [1].

[1] Rice, John R., A Metalgorithm for Adaptive Quadrature, J. ACM to appear.

[2] ——————, Parallel Algorithms for Adaptive Quadrature - Convergence, Proc. IFIP '74 to appear.

* Work supported by Grant GP32940X from the National Science Foundation.

Richard F. Riesenfeld
University of Utah
Salt Lake City, Utah 84112

Computer Graphics as a Parallel/Associative Process

Computer graphics stands out as an area in which most of the basic algorithms exhibit a highly parallel nature in the sense that a single operation is typically performed on an entire data file representing an object for display. The most primitive manipulation transformations (translation, rotation, scaling, shearing, persepective) are effected by applying a 4 x 4 - matrix to each point of the object, a process which could usefully occur concurrently. Furthermore, the data file can often be generated in parallel if it is derived from a closed form equation like a Coons patch, or a B-spline surface. Similarly Gouraud/Phong-type shading lends itself to parallel computation. Even the recursive algorithms for surface rendering like Warnock's or Catmull's Algorithm could run as a parallel instead of a serial process.

Other standard graphics algorithms like hidden surface removal and clipping can be stated in terms of associative processes: Flag all polygons in a file that possess property P.

The most immediate conclusion is that the advent of parallel processors could have a profound effect on the formulation of computer graphics algorithms. Moreover, since real-time interactive graphics is still on the fringes of present hardware capabilities, parallel machines could easily be used to make real-time response a common phenomenon. Finally, a "graphics language" that allows the user to specify the parallel and associative operations as simple statements would be needed to make convenient use of these features. In short, computer graphics is very amenable to parallel processing.

Ahmed H. Sameh and David J. Kuck

Center for Advanced Computation
and Computer Science Department
University of Illinois
Urbana, Illinois 61801

A Parallel QR-Algorithm for Symmetric Tridiagonal Matrices

An iteration of the QR-algorithm for a symmetric tridiagonal matrix A may be described as follows:

(a) for a suitably chosen origin shift $k_r (r=1,2,\ldots)$ we reduce the matrix $(A_r - k_r I)$, where $A_1 \equiv A$, to the upper triangular form using an orthogonal transformation Q_r,

$$Q_r(A_r - k_r I) = R_r \qquad r = 1,2,\ldots$$

(b) the next iterate A_{r+1} is then given by

$$A_{r+1} = R_r Q_r^t$$

so that the symmetric tridiagonal matrix A_{r+1} is similar to $(A_r - k_r I)$, i.e., similar to $(A_1 - \sum_{i=1}^{r} k_i I)$ rather than A_1.

Assuming that A_r is of order n, it has been shown that the above iteration on a serial computer requires 4n additions, 4n multiplications, 3 divisions, and no square roots.

In this paper we are concerned with parallel computation, and assume that our parallel computer satisfies the following:

(i) any number of processors may be used at any time, but we will give bounds on this number.

(ii) each processor may perform any of the four arithmetic operations in one time step.

(iii) there are no memory or data alignment time penalties.

The orthogonal matrix Q_r is obtained as the product of (n-1) Givens transformations. Assuming that $A_r \equiv [\beta_i, \alpha_i, \beta_{i+1}]_{nxn}$, then Givens transformations are given by,

$$s_i = \beta_{i+1}/\gamma_i$$

$$c_i = (\alpha_i c_{i-1} - \beta_i s_{i-1} c_{i-2})/\gamma_i \qquad i = 1,2,\ldots,n-1$$

where $s_0 = 0$, $c_0 = 1$. We show that c_i and s_i can be obtained by solving two successive linear recurrence relations of the second and first orders, respectively. Thus, part (a) of the iteration requires $(8 + 5 \log_2 n)$ steps using no more than 4n processors. Similarly, we show that part (b) requires $(5 + \log_2 n)$ steps and two square roots using no more than (2n-1) processors. Hence the total parallel computation time of the iteration is, $T_p = (13 + 6 \log_2 n)$ steps with $p = 4n$ processors. Since the serial time $T_1 = 11n$, then the speedup over the serial computation is given by,

$$S_p = T_1/T_p \simeq 2n/\log_2 n$$

with an efficiency of,

$$E_p = S_p/p \simeq 1/(2 \log_2 n).$$

Ahmed Sameh and Terry Layman
Center for Advanced Computation
University of Illinois
Urbana, Illinois 61801

Toward an ILLIAC IV Library

An effort for writing a collection of numerical algorithms suitable for Illiac IV has been going on for more than two years at the Center for Advanced Computation at the University of Illinois. This collection of some 20 algorithms was intended to serve as a nucleus for an Illiac IV scientific subroutine library. These algorithms, which are coded in either Illiac IV assembly language ASK or the higher level language GLYPNIR, have all been checked out in a limited way on the Illiac IV simulator (on the Burroughs B6700 at UCSD in San Diego). Only limited checking could be done because of the extremely slow rate of simulated execution. Testing of these codes on Illiac IV itself has only recently begun.

Algorithms designed for (mostly core-contained) problems in the following application areas have been developed:

A. Solving systems of linear equations
B. Finding the eigenvalues and eigenvectors of real, symmetric tri-diagonal matrices
C. Solving a limited class of elliptic P.D.E.s
D. Computing Fast Fourier Transform
E. Solving Poisson's equation on a rectangular region
F. Finding the real roots of real polynomials
G. Linear Programming

In this paper we discuss the implementation of four routines for the eigenvalue problem:

1. A parallel Jacobi method for finding all the eigenvalues and eigenvectors of a real symmetric matrix (core-contained). [JACOBI]-GLYPNIR
2. A multisectioning method for finding the eigenvalues of a large symmetric tridiagonal matrix (core-contained). [MULTISEC]-GLYPNIR
3. A routine to find the eigenvectors corresponding to given eigenvalues of a large symmetric tri-diagonal matrix (core-contained). [TRIVEC]-GLYPNIR
4. A combination of a slightly modified JACOBI and a routine to reduce a real matrix ($N \leq 128$) to a normal matrix using similarity transforms ([EBER]-GLYPNIR) which calculates the eigenvalues and eigenvectors of real nonsymmetric matrices (core-contained).

Furthermore, we will present results of several test programs on the Illiac IV itself, and timing comparisons between these parallel routines and the equivalent serial algorithms, namely those ALGOL programs edited by Wilkinson and Reinsch in the second volume of the Handbook for Automatic Computation.

James R. Schiess
Laura C. Jackson
NASA Langley Research Center
Hampton, Virginia 23665

Sequential Filtering Algorithms for the STAR Computer

One of the major problems in modern engineering and science is the estimation of the state variables of a mathematical model of an observed dynamical process. For such problems, large quantities of measured data are often processed by a filtering scheme; additionally, the mathematical model may be a highly complex one containing dozens of variables. Because of the possible enormity of such problems, a vector processing computer, such as the STAR computer, provides a potential means for decreasing the computational effort and time required to process data and arrive at an acceptable solution.

In the present paper attention is restricted to adaptation of currently available linear filtering schemes to vector processing computers. Since it is standard practice to linearize nonlinear mathematical models of dynamical systems before applying filtering techniques, the conventional notation of linear models is briefly presented. The Kalman-Bucy filter is introduced as the standard filtering method, and two modifications of this method and the Potter "square root" filter are presented as alternative approaches. Efficient implementation of the individual stages of these filters on a vector processing computer is considered with emphasis placed on the number of required operations. Although on conventional scalar computers the Potter filter requires 30% more central processor time than the Kalman-Bucy filter, the analysis indicates the Potter filter is a competitive alternative on a vector processing computer. This is particularly advantageous since the Potter filter was designed to avoid certain divergence problems which arise when the Kalman-Bucy filter is applied to large amounts of data.

Harold S. Stone
University of Massachusetts
Amherst, Massachusetts 01002

The Structure and Use of Array Computers

The array computer is much like a conventional computer except that it operates on vectors of data in the way that a conventional computer operates on single items. One of the foremost examples of the array computer is the ILLIAC IV computer, now installed at NASA Ames Research Center. It has 64 separate processors, and thus operates naturally on vectors of length 64 or multiples thereof.

The architecture of the array computer imposes severe constraints on programs if they are to run efficiently. Presently it is known that some problems can be solved very efficiently, and that there exist no methods whatsoever to solve others efficiently. By far the largest class of problems are those for which it is still uncertain how to solve them efficiently with an array computer or if indeed a method exists.

For an array computer to be effective on any particular problem, the problem must satisfy at least the three following characteristics:

1. The computation must be describable as a sequence of vector operations. That is, all arithmetic operations performed at any given moment are identical.

2. If the computation requires data to be communicated among processors, then the data flow must be supportable by the existing processor-to-processor connections.

3. When several operands must be manipulated simultaneously they must be stored in different processors to prevent memory conflicts when they are accessed.

We explore some features of the ILLIAC IV and some techniques for programming array processors that have greatly enlarged the class of problems that satisfy these three constraints.

John C. Thompson
Computing Sciences Department
University of Oklahoma
905 Asp Avenue
Norman, Oklahoma 73069

Implementation of Sophisticated Constitutive Equations on the ILLIAC IV

There are a number of problems concerned with the simulation of ground motion response to explosive loading whose running time on current serial machines is excessive. These simulation programs which use finite difference and finite element techniques to integrate the equations of motion also require sophisticated equations of state to model media response. The programming of these constitutive relations on a parallel machine is challanging because in general, the state of the material depends on the current value of one or more deformation variables such as strain as well as the past stress history of the material. Therefore to calculate the current state of stress using the full ILLIAC IV Array, we must consider some cells to be loading while others are unloading. In the materials of interest, these two types of behavior obey different stress-strain relations.

The particular constitutive models studied in this paper were: a general non-linear hysteretic material, the variable modulus model of Nelson, the Cap model of Demaggio and Sandler, and a brittle failure model suggested by Cooper and the author.

To develop parallel algorithms for all of these models, each phase of the material behavior was written in the same functional form so that, for the most part, different material states were distinguished by a choice of parameter values rather than different logic paths. The language used was CFD. General estimates of the efficiency of each program and the speed increase resulting from parallelism are given.

S. M. Yen, D. S. Watanabe, and J. R. Flood
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Implementation of Finite Difference Schemes for Solving Fluid Dynamics Problems on ILLIAC IV *

We have used a conventional computer to study storage schemes and the treatment of joints that occur when more than 64 nodes are used, methods of minimizing the number of arithmetic operations required in parallel computations, methods of exploiting the parallelism implicit in any difference scheme, and methods for efficiently handling boundary conditions. We have examined one-dimensional problems which are useful in determining the effectiveness of parallel computational strategies.

In a problem with more than 64 nodes, we must devise appropriate storage schemes and optimize the evaluation of arithmetic expressions. In considering the interior calculation, straight storage seems most appropriate. We can use indexed addressing and routing to fetch any desired vector operand. A sample ILLIAC IV code to illustrate this process was written.

ILLIAC IV is a vector processor that operates on 64 element vector operands and can perform the usual arithmetic binary operations. The machine is operating "efficiently" when each binary operation produces a vector, most of whose elements were not available before and which are useful. To facilitate our study of a given difference scheme, we associate a graph that symbolically represents the sequence of arithmetic operations required by this formulation. By examining the graph, we can determine whether the formulation can be efficiently implemented on the ILLIAC IV. This approach allows us to evaluate schemes without writing actual codes.

We have analyzed several difference schemes and have found that all of these schemes can be implemented efficiently if they are cast in a multilevel formulation. Redundant computations are eliminated if the operations required at a given level are performed for the whole flow field before proceeding to the operations at the next level. However, this process requires extra storage. In multi-dimensional problems, the trade off between this extra storage and the elimination of redundant computations must be examined carefully.

A problem which occurs with any size blocks of nodes is the treatment of nodes near the boundaries. Suppose we have boundary conditions fixing values of dependent variables and we are using a difference scheme which is so "wide" that it cannot be used near the boundary. Suppose we use a scheme for these points that is of the same complexity as the interior scheme. Then in the worst case, if each scheme requires n operations, we need 2n operations to handle the boundary point and the next 63 interior nodes. We wish to find schemes for near boundary points that can be "imbedded" in the interior scheme so that both can be evaluated in one instruction stream of length $n + m$, $m \ll n$.

We studied two "imbedded" or hybrid schemes by examining their performances on an one-dimensional flow produced by an accelerating piston whose position is given by $x_p(t) = t^3$. The hybrid schemes used were Cheng-Allen/MacCormack and Cheng-Allen/Predictor. The Cheng-Allen method was used for the interior nodes in each case. Cheng-Allen/MacCormack uses MacCormack's method near the boundaries. An analysis of the error norms indicated that the overall accuracy of the scheme was still second order, although the accuracy near the boundaries was degraded. Our results seem to indicate that in general the combination of schemes of the same order in this manner do not reduce the order of the method; however, the stability of the hybrid schemes may be affected. However, no tendency toward instability was observed in these tests. Although the results of the tests were satisfactory, the performance of the hybrid schemes did not achieve that of the MacCormack method alone. In fact, for the inviscid piston problem, the simpler formula and higher accuracy of the MacCormack method would outweigh any increase efficiency in computation for a hybrid method.

*Research supported by the Office of Naval Research under Contract N00014-67-A-0305-0019.